AD-A213 597

OCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 89-1282
6a. NAME OF PERFORMING ORGANIZATION: Carnegie-Mellon University 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: Air Force Office of Scientific Research
6c. ADDRESS (City, State, and ZIP Code) Intelligent Systems Laboratory Robotics Institute Pittsburgh, PA 15213	7b. ADDRESS (City, State, and ZIP Code) Building 410 Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR 8b. OFFICE SYMBOL (If applicable): NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F49620-85-C-0054
8c. ADDRESS (City, State, and ZIP Code) Building 410 Bolling AFB, DC 20332-6448	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 61102F PROJECT NO.: 2304 TASK NO: A7 WORK UNIT ACCESSION NO:

11. TITLE (Include Security Classification)
CONSTRAINT-BASED SCHEDULING IN AN INTELLIGENT LOGISTICS SUPPORT SYSTEM: AN ARTIFICIAL INTELLIGENCE APPROACH

12. PERSONAL AUTHOR(S)
Professor Stephen F. Smith

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 15Mar85 TO 14Mar86		

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report summarizes the research that was performed under AFOSR Contract Number F49620-85-C-0054 titled "Constraint-Based Scheduling in an Intelligent Logistics Support System: An Artificial Intelligence Approach", provided by the Electronics and Material Sciences Department of the Air Force. The overall goal of this research has been the development of a computational theory of constraint-directed scheduling for application to the problem of job shop production scheduling. Research performed under this contract has focused on the investigation of issues relating to the reactive management of job shop schedules in response to the dynamics of factory operation. An experimental knowledge-based system called OPIS (Opportunistic Intelligent Scheduler) has been developed that provides a scheduling system architecture for reactive control and a testbed for exploring different reactive scheduling methodologies. We provide an overview of this work and highlight the major accomplishments.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr Robert Buchal	22b. TELEPHONE (Include Area Code): (202) 767-4939 — 22c. OFFICE SYMBOL: NM

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

Constraint-Based Scheduling in an Intelligent Logistics Support System: An Artificial Intelligence Approach

Final Report for
~~supplemental grant to AFOSR Contract Number F49620-82-K-0017~~ corrected to read AFOSR Contract Number F49620-85-C-0054
provided by the Electronics and Material Sciences Department
of the Air Force

Stephen F. Smith

Intelligent Systems Laboratory
Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

15 January 1987

Abstract

This report summarizes the research that was performed under ~~a supplemental grant to~~ AFOSR Contract Number ~~F49620-82-K-0017~~ F49620-85-C-0054, titled "Constraint-Based Scheduling in an Intelligent Logistics Support System: An Artificial Intelligence Approach", provided by the Electronics and Material Sciences Department of the Air Force. The overall goal of this research has been the development of a computational theory of constraint-directed scheduling for application to the problem of job shop production scheduling. Research performed under ~~the supplemental grant~~ this contract has focused on the investigation of issues relating to the reactive management of job shop schedules in response to the dynamics of factory operation. An experimental knowledge-based system called OPIS (Opportunistic Intelligent Scheduler) has been developed that provides a scheduling system architecture for reactive control and a testbed for exploring different reactive scheduling methodologies. We provide an overview of this work and highlight the major accomplishments.

Table of Contents

A-1

List of Figures

1. Introduction

This report summarizes the research that was performed under a supplemental grant to AFOSR Contract Number F49620-82-K-0017, titled "Constraint-Based Scheduling in an Intelligent Logistics Support System: An Artificial Intelligence Approach", provided by the Electronics and Material Sciences Department of the Air Force. The overall goal of this research has been the development of Artificial Intelligence (AI) based theories and techniques that enable effective computer generated solutions to real world scheduling problems. The central thesis upon which this work is based is that solutions to realistic scheduling problems require a framework that enables consideration of all relevant scheduling constraints, and, further, that knowledge about the set of relevant constraints can provide significant leverage in formulating and maintaining good solutions. Thus, our research has sought to identify these sources of knowledge, and investigate their representation and use as the basis for a constraint-directed scheduling methodology. The difficult problem of job shop scheduling was chosen as the specific focus of the research, and it is the progress made toward solution of this problem that is the subject of this report.

Under the supplemental grant, our work has focused on issues relating to the reactive management of job shop schedules in response to the dynamics of factory operation. Building on the constraint-directed scheduling work performed during the original contract period, the following has been accomplished:

- A scheduling system architecture for conflict-directed control has been developed and implemented. The control architecture provides an integrated framework for predictive schedule generation (or expansion) and reactive schedule revision that is based on analysis of detected or predicted constraint conflicts. Central to this architecture is a coupling of techniques for constraint propagation and conflict detection with an event-based approach to representing and structuring of control knowledge.

- The opportunistic schedule generation methodology implemented and demonstrated in OPIS 0 [16, 14] has been generalized, and merged with an initial set of techniques for reactive schedule revision. These scheduling methods together with the system architecture define the OPIS 1 job shop scheduling system.

- A probabilistic framework for evaluating temporally imprecise abstract schedules with respect to expected resource usage has been developed. The framework has applications in the identification of bottleneck resources (a component of the opportunistic scheduling methodology just mentioned) and in developing abstract schedules over longer term time horizons.

The reader is referred to [16, 17, 18, 9, 10] for further accounts of this work.

The remainder of this report is organized as follows. In section 2, we provide a brief account of the work performed under the original contract period. In section 3 we summarize the research

performed under the supplemental grant. In Section 4, some final remarks are made.

2. Background

During the original contract period considerable progress was made toward development of a general theory of constraint-directed job shop scheduling. Much of this work revolved around the construction of several versions of a knowledge-based job shop scheduling system called ISIS (Intelligent Scheduling and Information System), each of which was tested with a model of the Westinghouse Electric Corporation Turbine Components Plant in Winston-Salem, NC. Investigations with ISIS led to the following contributions:

- Analysis of specific job shop environments resulted in an identification and categorization of the various types of constraints that influence scheduling decisions. This categorization includes organizational goals, causal restrictions, physical constraints, operational preferences, and resource availability constraints, and broadly delineates the knowledge requirements of a constraint-directed scheduling system.

- A frame-based representational framework for modeling all aspects of the manufacturing enterprise was developed. The framework ascribes a semantics to general concepts of activities, states, objects, causality, and time, which is then refined to provide primitives for modeling the production environment and its constraints. A central component of the framework is a constraint representation that extends predicate constraint specification techniques to enable the expression of preference constraints. This defines the notion of relaxable constraints (i.e. constraints that may be satisfied to varying degrees, depending on the specific situations of constraint conflict that arise during scheduling). The representation makes alternative choices explicit, and formalizes knowledge relating to preference relationships, constraint importance, constraint elasticity, constraint relevance, and interdependencies among constraints.

- A dynamic schedule evaluation scheme, based on knowledge defined in the constraint representation and intuitively reflecting how well the set of scheduling decisions under evaluation satisfies the relevant objectives and preferences, was developed. The evaluation scheme includes a methodology for constraint resolution (i.e. determination of precisely which constraints are relevant to a given scheduling decision) which is based on model semantics concerning placement of constraints in the model. The evaluation scheme is applicable in two contexts: it provides a basis for comparing alternative sets of scheduling decisions that are generated during the search for a good schedule, and it provides a means for assessing the quality of user imposed scheduling decisions.

- Both generative and analytic techniques for intelligently compromising among conflicting constraints were defined. Generative (or search-based) relaxation utilizes the alternatives possible with respect to one constraint (e.g. the resources capable of performing a given operation) to generate a set of alternative decisions, each of which will variably relax other relevant constraints (e.g. due date constraint, resource preferences). Analytic (or rule-based) relaxation exploits knowledge relating to the importance of and interdependencies between various constraints to either restrict or redirect the search to specific areas of the solution space. These relaxation techniques were embedded within a hierarchical search architecture to provide an initial constraint-

directed schedule generation methodology. Assuming an order-based decomposition of the scheduling problem, the search architecture advocates the "staged introduction" of constraints in constructing a given order's schedule. This is accomplished by utilizing multiple levels of analysis, where the results of each level provide insight regarding the solution at lower levels.

More detailed accounts of this work may be found in [2, 3, 4, 5, 6, 7, 8, 15, 17].

As experience was accumulated with the ISIS scheduling architecture, weaknesses stemming from its strict reliance on an order-based decomposition of the problem were perceived. It was recognized that an a priori commitment to a single "scheduling perspective" introduced a bias with respect to the types of constraint conflicts that could be effectively resolved, in this case resulting in poor satisfaction of constraints surrounding the allocation of specific resources (e.g. sequencing preferences to minimize machine setup changes) [19]. It was hypothesized that a scheduler must be capable of selectively adopting different scheduling perspectives (i.e. selectively decomposing the problem in different ways) to effectively attend to the full range of constraints.

To provide experimental justification for this claim, an initial multi-perspective scheduling system including both resource-based and order-based scheduling perspectives was configured. A resource scheduling method based on the selective use of a set of dispatch scheduling heuristics was implemented, and the scheduling method of ISIS was adapted for use as the order scheduling method. An assumption was made that the important resource-centered constraints were those that surrounded allocation of the bottleneck resources, and to simplify issues of coordination, the following, tightly controlled pattern of interaction between these two scheduling perspectives was imposed:

- The resource scheduler was first applied to a single, pre-specified bottleneck resource (a work area of the plant consisting of some number of machines).
- The order scheduler was then applied to work outward from this established portion of the shop schedule to complete the schedules for each individual order.

The performance of this system, designated OPIS 0, was then contrasted with that of ISIS and a dispatch system using the COVERT priority rule for minimizing tardiness cost (as formulated in [20]). The latter system represents a well known and well regarded approach to job shop scheduling, and was included to provide a benchmark for the experimental study. In the experimental study, OPIS 0 was seen to significantly outperform both single perspective systems with respect to minimizing tardiness cost, minimizing work-in-process time and minimizing machine setup changes (the three performance objectives that were considered). More detailed descriptions of OPIS 0 may be found in [12, 13, 14, 16]. A complete account of the experimental study may be found in [11].

3. Research Summary

Generally speaking, the problems which led to the configuration of OPIS 0 reflect a need to reason about constraints and constraint relaxation at higher levels. The research conducted with ISIS emphasized mechanisms for constraint-directed reasoning at the "micro" (or individual decision) level. However, many global search control decisions (such as those relating to how the problem should be decomposed) affect the system's ability to satisfy classes of constraints, and hence must be based on more "macro" level analyses of the problem constraints; in particular an ability to recognize important constraint conflicts (or types of conflicts) and direct problem solving activity accordingly is essential to effective use of "micro" level constraint relaxation techniques. OPIS 0 demonstrates the utility of such a *conflict-directed* approach to structuring the search for a good solution.

At the same time, the OPIS 0 implementation imposes some rather severe limitations with respect to system flexibility:

- A rather obvious restriction in OPIS 0 is its reliance on a static decomposition of the scheduling problem. A single pre-specified bottleneck resource is used to drive a fixed high level strategy for partitioning effort between the two scheduling perspectives. In an actual workshop, the situation is often much more complex. Several bottleneck resources may exist, either independently of one another or in a specific primary bottleneck/secondary bottleneck relationship. Furthermore, the bottleneck resource in the shop often "floats" over time, in which case specific resources need not be considered critical for the entire duration of the schedule. A priori specification of these more complex resource requirements is unreasonable, as many of the specific relationships emerge only during the scheduling activity (i.e. once some number of scheduling decisions have been made). An ability to dynamically predict "high contention" areas of the shop schedule is necessary to fully exploit the resource based scheduling perspective.

- A second limitation of OPIS 0 concerns its strict assumption that the resource-based subproblem at the bottleneck resource completely dominates the order-based subproblems that involve non-bottleneck operations. The bottleneck schedule that is generated by the resource scheduler is guaranteed to be feasible (i.e. at least one conflict-free shop schedule is realizable), and the order scheduler is obliged to generate scheduling decisions that are consistent with the bottleneck schedule. This guarantee of feasibility is accomplished within the resource scheduler by actually building and maintaining a tentative (albeit simple) schedule for all resources required to perform operations that must proceed the bottleneck operations. These tentative schedules are discarded once the bottleneck resource schedules have been finalized. There are two reasons why this is undesirable. First, the guarantee of feasibility is not exacted without the computational expense associated with generating the tentative schedules. More importantly, however, important concerns are likely to arise in subsequent subproblems which should force reconsideration of the existing bottleneck schedule (and subsequent compromise). The subproblem dominance assumptions made in OPIS 0 preclude this possibility.

- A final limitation of OPIS 0, perhaps a more general statement of the limitation just described, is that it implements a schedule generation strategy and, as such, is insensitive to the dynamics of the shop floor. Unanticipated events (e.g. machine breakdowns, power failures) are commonplace on the shop floor, and continually introduce conflicts into the current shop schedule. Short of regenerating the entire shop schedule (which is obviously not often the desirable course of action), OPIS 0 has no capabilities for reacting to such events. At the same time, however, the attractiveness of multiple scheduling perspectives in responding to unanticipated events is fairly clear. There are some events that suggest a resource-centered perspective (e.g. a machine breakdown) while there are others that are more effectively addressed from an order-based perspective (e.g. a request for rework with respect to an in-process order).

The research performed under the supplemental grant period has sought to address these issues. It has led to development of the OPIS 1 scheduling system (here after referred to simply as OPIS) which defines a general framework for conflict-directed control, extends the OPIS 0 scheduling capabilities (particularly in the area of reactive schedule maintenance), and provides a testbed for exploring additional scheduling and rescheduling strategies. It has also resulted in an approach to reasoning with less precise schedules, which is necessary to effectively maintain schedules over longer term time horizons. The major accomplishments of this research are summarized in the subsections below.

3.1. A Scheduling Framework for Conflict-Directed Control

The limitations of OPIS 0 raised above all center around the need for greater system flexibility in approaching various scheduling tasks. In short, more dynamic and opportunistic control of problem decomposition and problem solving is required to fully address the scheduling requirements of actual factory environments. A loosening of the reins on opportunistic problem decomposition and schedule generation introduces considerable additional complexity into the problem solving process. One must give up the assumption that individual subproblem results (or solution components) will be compatible and admit the possibility that important local concerns will surface during the generation of specific solution components that lead to constraint violations when integrated with previously generated solution components. Thus, in addition to determining how the problem should be decomposed and in which order various components of the schedule should be generated, the scheduler must be capable of monitoring progress made toward a final schedule, recognizing and characterizing conflicts in the schedule as they arise, and using these characterizations to initiate appropriate schedule revision activities.[1] Note, however, that these capabilities also provide the

[1] Given the granularity of the solution components that are being synthesized (e.g. a schedule for a specific work area) and the high degree of interdependency among the decisions that comprise these solution components (due to the temporal and resource constraints on the problem), systematic backtracking procedures are of little use in resolving conflicts. The system's approach to revision must be driven by characteristics of the conflict (or conflicts) at hand.

necessary machinery for responding to unanticipated external events. This process differs only in the fact that conflicts are introduced into the predictive schedule through indications that the status of the factory has changed. The OPIS control architecture described below provides these capabilities, and, in doing so, provides a framework that merges the activities of predictive schedule generation/expansion and reactive schedule maintenance. Further details may be found in [18].

3.1.1. Overview

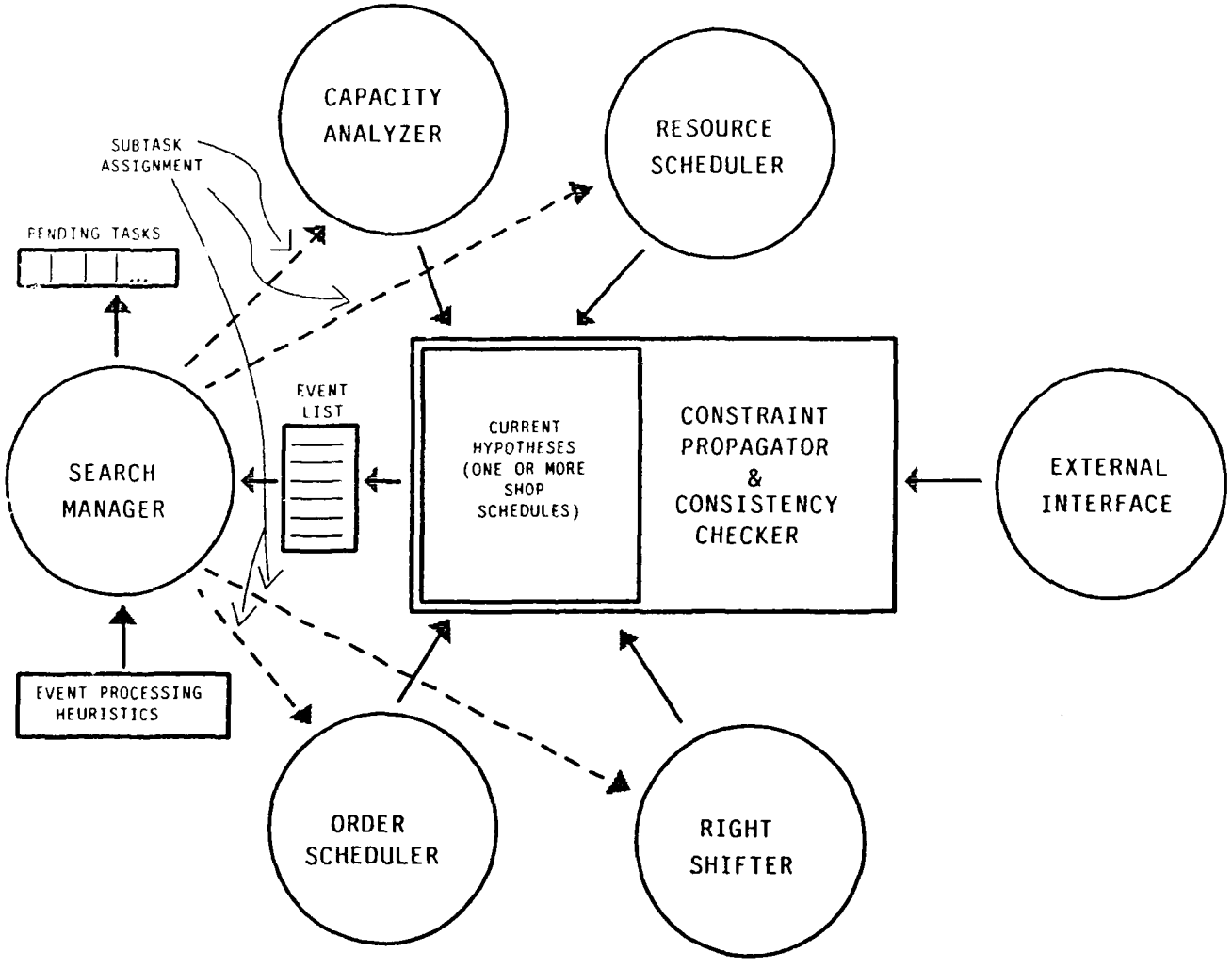

Figure 3-1: Current OPIS Architecture: Top Level

Figure 3-1 depicts the top level structure of OPIS, and identifies the major components of the

current system architecture. The organization is a variation of the HEARSAY-II blackboard style architecture [1], and similarly assumes a system organization comprised of a number of knowledge sources (KSs) that extend and revise a global set of one or more hypotheses. In this case, the KSs implement alternative scheduling methods and the hypotheses being manipulated are candidate shop schedules. For simplicity in the following discussion we will assume that only a single shop schedule is being manipulated.[2]

Within this architecture, a designated KS called the manager assumes responsibility for planning and coordinating the scheduling effort. Scheduling proceeds via the formulation and initiation of *scheduling tasks*. Each scheduling task requests a particular analysis, extension or revision relative to the current shop schedule (e.g. generate a schedule for work area *wa-1*, revise the schedule for order *ord-1*, analyze the capacity of the shop), and designates a specific scheduling KS to carry out the task. The manager's queue of *pending subtasks* constitutes its current plan for solving the scheduling problem at hand. The execution of a scheduling tasks by scheduling KSs yields changes to current shop schedule. These changes are integrated into the current hypothesis by the *schedule management subsystem*, which exploits the temporal restrictions and capacity limitations specified in the factory model to determine the additional constraints imposed on the schedule by each new scheduling decision. This provides an accurate characterization of the current state of the evolving solution and a straightforward basis for detecting conflicts. The manager is informed of the results of KS execution through the posting of *control events*, which summarize those aspects of KS execution that are of importance from a control perspective. Events may highlight important characteristics of the current solution constraints (e.g. resource contention is likely to be high at a particular resource), indicate that progress has been made toward a final solution (e.g. another component of the shop schedule has been generated), or identify specific problems that are present the solution (e.g. constraint violations). The manager responds to posted events through application of a set of *event processing heuristics*. This results in the formulation of new scheduling tasks, and the queue of pending subtasks is updated accordingly. Thus, the manager implements an event-driven control regime, continually revising its "scheduling plan" as the results of KS execution become known.

At the core of this framework for control are two key notions:

---

[2]Note that this does not mean that there is no search taking place; but rather that exploration of alternative scheduling decisions is confined to the local subproblems addressed by individual KSs. For example, in constructing a given order's schedule the order scheduler will conduct a search before committing to a particular set of scheduling decisions. Once this commitment has been made, however, it becomes part of a single evolving shop schedule. This "single shop schedule" assumption is necessary in the context of schedule generation to keep the problem computationally tractable. The provision for maintaining multiple shop schedules is included in the OPIS architecture primarily for reacting to external events. Here, the problem is smaller in scope, and it is quite feasible to consider alternative schedule revision strategies.

1. The use of a centralized schedule management component as a means of communicating constraints among subproblems and recognizing constraint violations, and

2. An event-based framework for representing and structuring search control knowledge.

These two notions are elaborated in the following subsections.

3.1.2. Schedule Management

The OPIS scheduling architecture makes no commitment as to the order in which individual scheduling decisions will be made. Rather, the architecture assumes that characteristics of the problem at hand, in particular analyses of the constraint conflicts that must be resolved, will be used to dynamically prioritize the scheduling decisions that have to be made. Thus, for example, schedule generation might proceed by constructing schedules for resources where contention is likely to be high and then considering the scheduling decisions that involve the allocation of less critical resources. Similarly, indication that a machine will be down for the next week might lead to some amount of rescheduling in the work area containing the failed machine, followed by revision of any scheduling decisions in other parts of the schedule that are affected by this rescheduling. Essential to this opportunistic approach to scheduling is an ability to maintain an accurate characterization of the current state of the schedule. Both problem decomposition (the formulation of appropriate scheduling tasks) and subproblem solution (the application of specific scheduling KSs) require knowledge of the constraints that are currently imposed on the schedule (both by the current factory state and the scheduling decisions that have already been made).

This support is provided within OPIS by incrementally maintaining an explicit representation of (1) the current temporal constraints on each manufacturing operation that must be scheduled, and (2) the current availability of each resource. Thus, an operation description delineates, at any point, the set of allocation decisions that are compatible with the constraints imposed by the external world and any other scheduling decisions that have already been made. This is illustrated by the partial description in Figure 3-2, which states that operation **ord2-P1-root-grinding** must be scheduled in the **208-rooting-area** sometime between 10:30 and 17:00 on August 18th to remain consistent with the current solution. In this case, the end time constraint is a consequence of the scheduling decision that was made concerning the downstream operation **ord2-P1-milling-operation**, and the start time constraint is a consequence of both the order's release date and unavailability of the 208 rooting machines (perhaps due to their prior allocation to other operations). Descriptions of resource availability, which are associated with the resources themselves, characterize intervals of time during which the resource may still be allocated and how much available capacity remains (in the case of aggregate resources). These representations of current time and capacity constraints are maintained

at different levels of aggregation to enable scheduling at various levels of precision.

{{ord2-P1-root-grinding
{INSTANCE: **P1-root-grinding-operation**
ORDER: **order2**
RESOURCE-REQUIREMENTS: **208-rooting-area**
NEXT-OPERATION: **ord2-P1-milling-operation**
STATUS: unscheduled
TIME-BOUND-INTERVAL: {{INSTANCE: **calendar-time-interval**
START-TIME: Aug 18 10:30
origins: (order-release-date **order2**)
(capacity-restriction **208-rooting-area**)
END-TIME: Aug 18 17:00
origins: (scheduled-start-time **ord2-P1-milling-operation**)
}} } }}

Figure 3-2: An unscheduled operation with time bound constraints as represented in the OPIS knowledge base

---

These constraint representations are maintained by a set of propagation processes collectively referred to as the schedule management subsystem. The propagation processes are driven by the temporal restrictions (e.g. operation precedences, operation durations) and resource requirements specified in the factory model, and are triggered whenever changes to the schedule are made. When operations are scheduled, for example, their descriptions are updated to reflect the chosen resources and intervals of execution, and constraints are propagated to both related operations and the resources that have been allocated. Changes to the schedule may be more complex than the addition or retraction of individual scheduling decisions. For example, indication that an order must be reworked requires the addition of the necessary rework operations to the production plan before time bound propagation can be carried out. Specific "schedule update" processes are defined for each type of change to the schedule that might be encountered.

As mentioned previously, this explicit representation of the current state of the schedule serves two purposes within the OPIS scheduling architecture. The first is one of constraint communication, making all current constraints apparent to scheduling KSs during generation or revision of specific components of the overall schedule. The second is in providing a basis for detecting problems in the schedule. The representation we have described enables straightforward detection of three basic types of constraint conflicts: *time* conflicts (corresponding to precedence violations) *capacity* conflicts (corresponding to resource availability violations), and *time vs capacity* conflicts

(corresponding to situations where scheduling decisions do not exist that mutually satisfy current temporal restrictions and resource availability constraints). In OPIS, detection of these types of conflicts is coupled with the constraint propagation processes, and each conflict detected is posted with the manager (see below). Complete details of the constraint propagation processes and the detection of conflicts may be found in [9].

3.1.3. Event-Based Control

The OPIS manager formulates, extends, and revises its current scheduling plan (i.e. its queue of pending subtasks) in response to posted *control events*. Control events represent those consequences of internally initiated scheduling actions (i.e. KS execution) and externally initiated schedule updates that are relevant to control decisions. They are generated and posted as a result of either activity. Events provide an abstract view of the current state of the schedule, and contain all the information necessary for the manager to determine how to proceed. Events also provide a means for organizing the system's control knowledge. *Event processing heuristics*, which map occurrences of particular events to appropriate sequences of scheduling tasks, and knowledge relating to event importance, which is used in ordering the queue of pending subtasks, are directly attached to the prototype description of each event type and therefore directly accessible to the manager in responding to specific events. These ideas are illustrated in Figure 3-3.

{{**precedence-violation**
{IS-A: **elementary-conflict**
CONFLICTING-COMMITMENTS:
HYPOTHESIS:
MAGNITUDE:
INTRODUCED-BY:
EVENT-TYPE-IMPORTANCE: **4**
EVENT-PROCESSING-HEURISTICS: **pv-heur1 pv-heur2**
process-event: process-event
calculate-overall-significance: calc-pv-significance
verify-control-state: pv-state-check } }}

Figure 3-3: The **precedence-violation** event prototype

The set of control events defined within the OPIS architecture are categorized into three general classes:

- *conflicts* - Conflict events are used to characterize inconsistent sets of scheduling decisions that have been detected in the schedule. Such events involve the violation of non-negotiable constraints and are precisely those that are detectable by the schedule

management subsystem. Reaction in this case is mandatory as the current schedule is infeasible.

- *compromises* - Compromise events are produced as the result of analysis tasks and concern the violation of preference constraints. Two event subtypes are distinguished here: *unsatisfactory compromises*, which identify preference concerns that have been unacceptably relaxed, and *predicted compromises*, which designate areas in the schedule where it appears that it will be necessary to compromise preferences. Unsatisfactory compromise events may or may not be reacted to, depending on the manager's perception of the opportunities for improvement. Predicted compromises provide a basis for prioritizing the set of scheduling decisions that must be made.

- *hypothesis-modifications* - Hypothesis modification events simply indicate changes that have been made to the current schedule. They are posted as a result of either KS execution or factory status updates. In the former context, hypothesis modification events provide a basis for stringing together specific sets of subtask creation heuristics to implement particular schedule development strategies.

Upon initial invocation and at the end of each top-level problem solving step, the OPIS manager responds to the currently posted set of events. This activity proceeds in two steps:

1. *event aggregation*, during which the most appropriate set of events to consider is determined, and

2. *event processing*, during which the manager's current scheduling plan is revised in response to this determined set of events.

During the event aggregation step, any "related" events among those that have been posted are combined into aggregate events. Because of the fact that KS execution and external status updates may result in a considerable amount of change to the current solution (e.g. the resource scheduler may make decisions for all orders that pass through a given work area), several constraint conflicts can be introduced during a single top-level problem solving step. As we have seen, each conflict is detected and reported individually during constraint propagation. It is often the case that these elementary events are related in some manner (e.g. they involve different operations of the same order, they involve different operations requiring the same resource, etc.), and would be better addressed by the system simultaneously. Thus, the notion of an aggregate event is introduced, and aggregate event types are defined on the basis of such relationships. *Event aggregation heuristics* are associated with these descriptions to specify the precise circumstances under which two or more posted events should be transformed into an aggregate event of a given type.

During the event processing step, the event processing heuristics associated with each currently posted event (including any aggregate events generated during the event aggregation step) are applied to determine how the system should proceed. These condition/action rules examine

characteristics of the event being processed and specify extensions and revisions to the manager's current queue of pending subtasks. These changes involve some combination of the following primitive actions: the creation of new subtasks to perform, a reordering of existing tasks in the queue, and the elimination of existing tasks. Once all events have been processed, the queue of pending subtasks is updated and the highest priority pending subtask is initiated. Subtask prioritization is a function of the significance of the triggering event type (e.g. tasks resulting from conflict events are generally considered more important than those resulting from hypothesis modification events), characteristics of the triggering event (e.g. the magnitude of the conflict reported), and characteristics of the task itself (e.g. its dependencies with respect to other pending subtasks).

3.2. Scheduling Methods and Strategies

Four scheduling KSs have been implemented within the architectural framework described above (see Figure 3-1). These KSs provide a basic set of methods which can be combined in different ways to produce different scheduling and rescheduling strategies. They include:

- order scheduler KS - The order scheduler implements the order scheduling method used within OPIS 0 (essentially the detailed resource analysis and resource assignment levels of ISIS - see [7]), revised to operate with the propagated time bound constraints. It can be invoked to either generate or revise scheduling decisions for a designated sequence of operations for a given order. When invoked in a reactive mode (i.e. to revise decisions in response to detected problems), previous decisions are treated as additional preference constraints on the decisions to be made.

- resource scheduler KS - The resource scheduler includes the resource schedule generation method used in OPIS 0 (see [13]) as well as a newly developed capability for revising an existing resource schedule. Thus, like the order scheduler, it can be invoked in either predictive or reactive contexts, in this case relative to the scheduling decisions that involve a designated resource (i.e. machine or work area). The approach to schedule revision attempts to retract only as many scheduling decisions as necessary to produce a new, conflict-free schedule for the designated resource. This is accomplished by assuming no schedule forward in time from the point of the current problem (e.g. order contention due to insufficient capacity), and invoking the schedule generation strategy. However, after each generation step, which adds one or more additional operations to the designated resource's schedule, an analysis is performed to see if the new schedule can be consistently synthesized with the fragment of the old schedule consisting of the operations that have yet to be placed in the new schedule.

- capacity analyzer KS - This KS implements a "shop level" scheduling perspective. It provides a basis for dynamic problem decomposition by generating predictions of likely areas of high resource contention. In contrast to the detailed schedulers, the capacity analyzer operates with aggregate descriptions of resources, operations and resource allocation decisions. It constructs a predictive shop schedule that satisfies the time bound constraints posted with each aggregated operation, using a general line balancing heuristic. The demand for capacity reflected by this schedule is then compared with the actual capacity of the required aggregate resources and likely bottlenecks are predicted.

- schedule shifter KS - This KS implements a simple "schedule shifting" strategy which simply moves the scheduled execution times of designated operations forward or backward in time. It is employed to resolve minor inconsistencies that arise in the schedule.

The manager's current body of control heuristics generalizes the OPIS 0 schedule generation strategy to one where effort is initially focused on scheduling any number of predicted bottlenecks (as dynamically determined by the capacity analyzer). The schedule is then completed on an order by order basis, and contingencies are included for revising decisions made by the resource scheduler in response to inconsistencies that arise later on in the search. Heuristics are also in place for reactively revising the current schedule as status updates are received from the factory floor. These heuristics define strategies for responding to machine failures, operation delays, rework requests, and the receipt of new orders.

3.3. Probabilistic Capacity Analysis

In integrating predictive schedule generation/expansion with reactive schedule revision, an important issue that arises concerns the level of detail at which scheduling decisions should be made and maintained. We have implicitly assumed in our discussion thus far that the goal is generation and maintenance of detailed production schedules. However, the level of precision at which scheduling decisions are made must be tied to their susceptibility to disruption. Given the unpredictability of the factory environment, the chances of executing a particular schedule become increasingly remote as the temporal distance to execution time increases. Thus, while detailed schedules are useful in the immediate short term time horizon, they are counterproductive over longer term time horizons. Not only does the generation of such plans require additional computational effort, but it also complicates later efforts to reactively maintain it. Less precise schedules, that can usefully guide schedule refinement as the external environment allows and requires it, must be maintained when scheduling over longer time horizons.

Some leverage can be gained by utilizing a hierarchical model and reasoning with more abstract problem descriptions over longer time horizons. As suggested in [17], abstract problems can be formulated in which operations to be scheduled require capacity on aggregate resources (representing functional groupings of individual resources) over their expected durations, where both the amount of capacity required by a given operation and its duration are derived from more detailed suboperation characteristics. However, as with the identity of the resource that will be used to perform a given operation, we would like to remain imprecise with respect to operation execution times, and this presents a difficult problem. Deterministic scheduling methods must make specific

allocation decisions (i.e. select specific operation execution times) in order to take resource capacity constraints into account. This defeats much of the purpose of abstracting in the first place since the resulting plans will designate a single point in the temporal dimension of the abstract solution space. We might consider generalizing these temporal constraints to delineate sets of possible allocation decisions after the fact. However, given the high degree of interaction between allocation decisions, it is difficult to imagine how this could be accomplished in any meaningful fashion. We might also consider reasoning with a coarser granularity of time at the aggregate level (e.g. time steps of hours instead of minutes). This approach is felt to offer some potential, but would also appear to introduce temporal imprecision in a fairly arbitrary fashion.

Addressing this need to remain temporally imprecise in the schedules generated over longer time horizons, we have developed a probabilistic framework for reasoning with sets of possible allocation decisions. More specifically, the framework provides a mechanism for evaluating an abstract, temporally underconstrained set of plans with respect to its expected requests for resource capacity. This is accomplished by developing a random model of the actual resource allocation process, and using the probabilistic characteristics of randomly generated allocation decisions as the basis for plan evaluation. Consistency constraints analogous to those that would result from a deterministic model of resource allocation can be defined, and the stochastic process can be biased to reflect the strategies and preference constraints of the actual deterministic allocator (i.e. the generator of final decisions for actual execution). In addition to providing a basis for "time bound scheduling" (i.e. the development of schedules that retain flexibility in operation execution times), the framework is also felt to provide the basis for a more accurate bottleneck detection mechanism. The reader is referred to [10] for a detailed account of this work.

4. Conclusions

In this report, we have summaried research that has addressed issues relating to the reactive management of factory schedules in response to the dynamics of factory operation. A scheduling system architecture has been defined that integrates previously developed schedule generation capabilities with newly developed techniques for reactive schedule revision in response to updates in factory status. More generally, the architecture provides an experimental testbed for exploring alternative reactive rescheduling methodologies. A probabilistic framework for reasoning about the resource requirements of temporally underconstrained schedules has also been developed, to provide a basis for generating and evaluating abstract schedules over longer term time horizons.

References

[1] Erman, L.D., Hayes-Roth, F., Lesser, V.R., and Reddy, D.R.
The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty.
Computing Surveys 12(2), June, 1980.

[2] Fox, M. S., B.P. Allen and G.A. Strohm.
Job Shop Scheduling: An Investigation in Constraint-Directed Reasoning.
In *Proceedings of the 2nd National Conference on Artificial Intelligence*, pages 155-158. American Association of Artificial Intelligence, August, 1982.

[3] Fox, M.S.
Constraint-Directed Search: A Case Study of Job Shop Scheduling.
PhD thesis, Computer Science Department, Carnegie-Mellon University, 1983.

[4] Fox, M.S., Smith, S.F., Allen, B.P., Strohm, G.A., and Wimberly, F.C.
ISIS: A Constraint-Directed Reasoning Approach to Job Shop Scheduling.
In *Proceedings IEEE Conference on Trends and Applications 83*. Gaithersberg, Maryland, May, 1983.

[5] Fox, M.S.
Knowledge Representation for Decision Support.
In Methlie, L.B. and Sprague, R.H. (editors), *Knowledge Representation for Decision Support Systems*. North-Holland, Amsterdam, 1985.

[6] Fox, M.S.
Observations on the Role of Constraints in Problem Solving.
In *Proceedings 6th Annual Conference of the Canadian Society for Computational Studies of Intelligence*. Montreal, Canada, July, 1986.

[7] Fox, M.S., and S.F. Smith.
ISIS: A Knowledge-Based System for Factory Scheduling.
Expert Systems 1(1):25-49, July, 1984.

[8] Fox, M.S. and S.F. Smith.
The Role of Intelligent Reactive Processing in Production Management.
In *Proceedings 13th Annual CAMI Technical Conference*. Clearwater, Florida, November, 1984.

[9] LePape, C. and S.F. Smith.
Management of Temporal Constraints for Factory Scheduling.
In *Proceedings AFCET Conference on Temporal Aspects in Information Systems*. North Holland, Paris, France, May, 1987.
also to appear as CMU Robotics Institute Technical Report.

[10] Muscettola, N. and S.F. Smith.
A Probabilistic Framework for Resource-Constrained Multi-Agent Planning.
Technical Report, The Robotics Institute, Carnegie Mellon University, forthcoming, 1987.

[11] Ow, P.S.
Experiments in Knowledge-Based Scheduling.
Technical Report, CMU Robotics Institute Technical Report, forthcoming, 1986.

[12] Ow, P.S. and S.F. Smith.
Toward an Opportunistic Scheduling System.
In *Proceedings 19th Hawaii International Conference on System Sciences*. January, 1986.

[13] Ow, P.S. and S.F. Smith.
Two Design Principles for Knowledge-Based Systems.
Decision Sciences , to appear, 1986.

[14] Ow, P.S. and S.F. Smith.
Viewing Scheduling as an Opportunistic Problem Solving Process.
In R.G. Jeroslow (editor), *Annals of Operations Research: Approaches to Intelligent Decision Support*. Baltzer Scientific Publishing Co., to appear, 1987.

[15] Smith, S.F.
Exploiting Temporal Knowledge to Organize Constraints.
Technical Report CMU-RI-TR-83-12, CMU Robotics Institute, July, 1983.

[16] Smith, S.F., P.S. Ow, C. LePape, B. McLaren, and N. Muscettola.
Integrating Multiple Scheduling Perspectives to Generate Detailed Production Plans.
In *Proceedings SME Conference on AI in Manufacturing*. Long Beach, CA, September, 1986.

[17] Smith, S.F., M.S. Fox, and P.S. Ow.
Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems.
AI Magazine 7(4), Fall, 1986.

[18] Smith, S.F., P.S. Ow, C. LePape, and N. Muscettola.
Reactive Management of Factory Schedules.
Technical Report, The Robotics Institute, Carnegie Mellon University, in preparation, 1986.

[19] Smith, S.F. and P.S. Ow.
The Use of Multiple Problem Decompositions in Time-Constrained Planning Tasks.
In *Proceedings 9th International Joint Conference on Artificial Intelligence*. Los Angeles, California, August, 1985.

[20] Ari Vepsaleinen.
State Dependent Priority Rules for Scheduling.
PhD thesis, Carnegie-Mellon University, April, 1984.